Date⇒ 25-01-2021
Module⇒ Backend
Lecture By⇒ Akash Handa
Subject ⇒ Introduction/Basics Of Backend

IN PREVIOUS LECTURE **(QUICK RECAP) Date-22/1/2021**	**In Today's Lecture (Overview)**
What is Npm? Installing npm Software Package Manager Managing Dependencies dependencies Dependency/Local Dependency Dev Dependency Global Dependency What Is Package.json What is Node.js? What is a Node.js File? What is a Module in Node.js?	What Is Nodemon Usage What Is Crud What is Express Node.js ExpressJS - Hello World Routers Route methods

What Is Nodemon

nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. ... **nodemon** is a replacement wrapper for node . To use **nodemon** , replace the word node on the command line when executing your script.
For Install

```
npm i nodemon
```

Usage

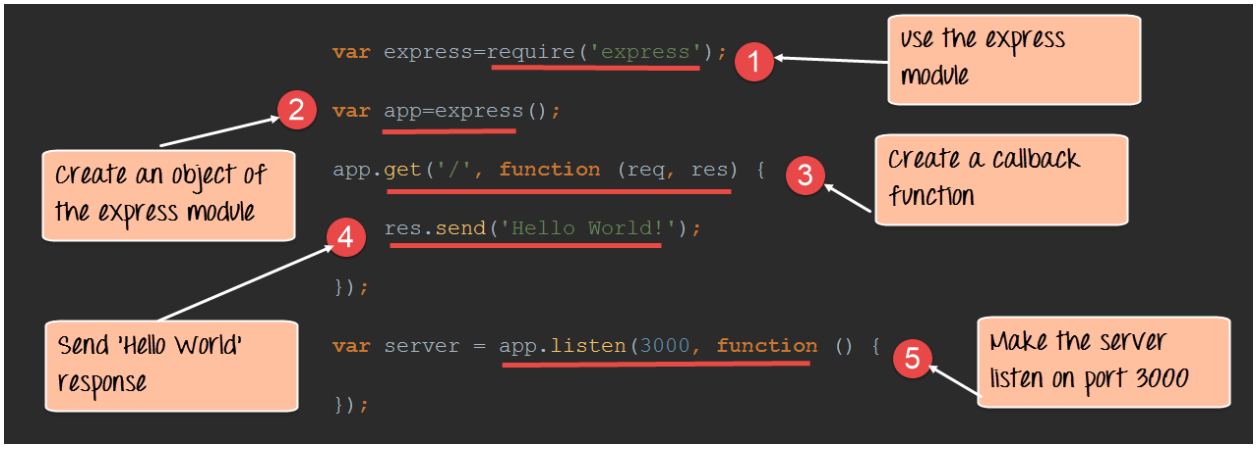
nodemon wraps your application, so you can pass all the arguments you would normally pass to your app:

nodemon [your node app]

What Is Crud

CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: create, read, update and delete.

What is Express Node.js



ExpressJS - Hello World

We have set up the development, now it is time to start developing our first app using Express. Create a new file called index.js and type the following in it.

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
   res.send("Hello world!");
});

app.listen(3000);
```

Save the file, go to your terminal and type the following.

```
nodemon index.js
```

Routers

Defining routes like above is very tedious to maintain. To separate the routes from our main index.js file, we will use Express.Router. Create a new file called things.js and type the following in it.

```
var express = require('express');
var router = express.Router();

router.get('/', function(req, res){
   res.send('GET route on things.');
});
router.post('/', function(req, res){
   res.send('POST route on things.');
});

//export this router to use in our index.js
module.exports = router;
```

Now to use this router in our index.js, type in the following before the app.listen function call.

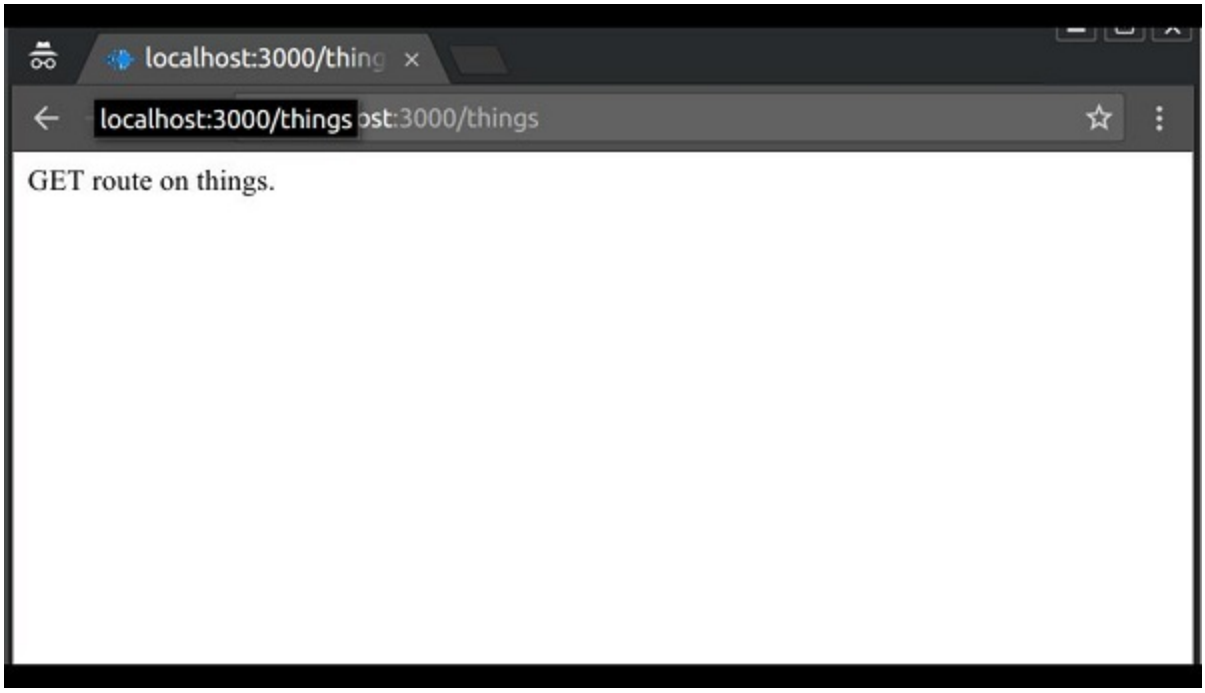
```
var express = require('Express');
var app = express();

var things = require('./things.js');

//both index.js and things.js should be in same directory
app.use('/things', things);

app.listen(3000);
```

The *app.use* function call on route '/things' attaches the things router with this route. Now whatever requests our app gets at the '/things', will be handled by our things.js router. The '/' route in things.js is actually a subroute of '/things'. Visit localhost:3000/things/ and you will see the following output.



We will look deeply into the Routing

routing refers to how an application's endpoints (URIs) respond to client requests. For an introduction to routing, see Basic routing.

You define routing using methods of the Express app object that correspond to HTTP methods; for example, app.get() to handle GET requests and app.post to handle POST requests. For a full list, see app.METHOD. You can also use app.all() to handle all HTTP methods and app.use() to specify middleware as the callback function (See Using middleware for details).

These routing methods specify a callback function (sometimes called "handler functions") called when the application receives a request to the specified route (endpoint) and HTTP method. In other words, the application "listens" for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

In fact, the routing methods can have more than one callback function as arguments. With multiple callback functions, it is important to provide next as an argument to the callback function and then call next() within the body of the function to hand off control to the next callback.

The following code is an example of a very basic route.

```
var express = require('express')
var app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
```

```
})
```

Route methods

A route method is derived from one of the HTTP methods, and is attached to an instance of the express class.

The following code is an example of routes that are defined for the GET and the POST methods to the root of the app.

```
// GET method route
app.get('/', function (req, res) {
    res.send('GET request to the homepage')
  })

  // POST method route
  app.post('/', function (req, res) {
    res.send('POST request to the homepage')
  })
```

Express supports methods that correspond to all HTTP request methods: get, post, and so on. For a full list, see app.METHOD.

There is a special routing method, app.all(), used to load middleware functions at a path for ***all*** HTTP request methods. For example, the following handler is executed for requests to the route “/secret” whether using GET, POST, PUT, DELETE, or any other HTTP request method supported in the http module.